REMARKS

Claims 1-2, 4-9, 11-23, and 25 are pending. Claims 3, 10, and 24 have been canceled, and claims 1, 4-5, 8, 11-12, 15-16, 20-22, and 25 have been amended. No new matter has been introduced. Reexamination and reconsideration of the present application are respectfully requested.

In the February 11, 2004 Office Action, the Examiner rejected claims 1-25. The Examiner rejected claims 1-25 under 35 U.S.C. 112, second paragraph as being indefinite for lacking antecedent basis in the independent claims. The Examiner rejected claims 1-3, 5-10, 12, 14,-16, and 18-24 under 35 U.S.C. § 102(b) as being anticipated by European Patent Application No. EP 0,747,831 to Neal ("the Neal reference"). The Examiner rejected claims 4, 11, 13, 17, and 25 under 35 U.S.C. § 103(a) as being unpatentable over Neal in view of Cataldo, "Intel Prepares for Server Chip Set Revival," EE Times, August 23, 2001 ("the Cataldo reference"). The Examiner's rejections are respectfully traversed.

The present invention is directed to an input/output hub that includes an inbound ordering queue (IOQ) to receive inbound transactions. All read and write transactions have a transaction completion. Peer to peer transactions are not permitted to reach a destination until after all prior writes in the IOQ have been completed. A write in a peer-to-peer transaction does not permit subsequent accesses to proceed until the write is guaranteed to be in an ordered domain of the destination. An IOQ read bypass buffer is provided to receive read transactions pushed from the IOQ to permit posted writes and read/write completions to progress through the IOQ. An outbound ordering queue (OOQ) stores outbound transactions and completions of the inbound transactions. The

OOQ also issues write completions for posted writes. An OOQ read bypass buffer is provided to receive read transactions pushed from the OOQ to permit posted writes and read/write completions to progress through the OOQ. An unordered domain within the input/output hub receives the inbound transactions transmitted by the IOQ and receives the outbound transactions transmitted by the unordered protocol. The unordered protocol may be a coherent interface.

Claim 1, as amended, recites:

An input/output hub, comprising:

an inbound ordering queue (IOQ) to receive inbound transactions, wherein all read and write transactions have a transaction completion, peer-to-peer transactions are not permitted to reach a destination until after all prior writes in the IOQ have been completed, and a write in a peer-to-peer transaction does not permit subsequent accesses to proceed until the write is guaranteed to be in an ordered domain of the destination;

an IOQ read bypass buffer to receive read transactions pushed from the IOQ to permit posted writes and read/write completions to progress through the IOQ;

an outbound ordering queue (OOQ) to store outbound transactions *from an unordered protocol, wherein the unordered protocol is a coherent interface,* and completions of the inbound transactions, and to issue a write completion for a posted write;

an OOQ read bypass buffer to receive read transactions pushed from the OOQ to permit the posted writes and the read/write completions to progress through the OOQ; and an unordered domain to receive the inbound transactions transmitted from the IOQ and to receive the outbound transactions transmitted from the_unordered protocol.

The Neal reference is directed to a data processing system which includes a host processor, a number of peripheral devices, and one or more bridges which may connect between the host, peripheral devices and other hosts or peripheral devices as in a network. Each bus to bus bridge connects between a primary bus and a secondary bus. Both the inbound path and the outbound path in the bus to bus bridge are controlled by a state machine which takes into consideration activity in both directions and permits or inhibits bypass transactions.

In the February 11 office action, in reference to the rejection of claim 3, the Examiner states that Neal "shows that the unordered protocol is a coherence interface (throughout the discussion of the system operation in cols. 3-7)." On March 8, 2004, during an Examiner's Interview, the Examiner was asked why he believed the Neal reference disclosed a coherent interface. (Notice the word was misspelled in the original claims. It was intended to be "coherent" not "coherence.") He stated that the entire reference disclosed a system where the packets arrive at their destination in the right order and in the fashion in which they were intended to arrive. Applicants' attorney then informed the Examiner that his definition was not what was intended by the word "coherent interface", and that coherent interface was used in the art to describe an interface that is specifically designed to support distributed multiprocessing and that entails accessing the cache of multiple processors in a certain way. The Examiner informed applicants' attorney that he was unaware of that definition and that the different definition should be noted in the amendment. A more lengthy description of a coherent interface can be found in the two articles attached herewith, both titled Scalable Coherent Interface, that describe the Scalable Coherent Interface (SCI), the IEEE standard for coherent interfaces.

The limitations of claim 3 have been added to claim 1 and claim 3 has been canceled. Claim 1, as amended, now recites the limitation wherein the unordered protocol is a coherent interface. Neal does not teach, suggest, or disclose a coherent interface with the above-mentioned definition, and thus, Applicants believe claim 1 distinguishes over the cited reference and is allowable.

The Examiner rejected dependent claim 8, stating "Neal also shows an intermediary device interconnecting the Producer-Consumer ordered interface and an input/output device (inside bridge 20 and as in fig.2)." During the March 8 interview, it was discovered that the Applicants had inserted the words "further including" in claim 8, when they had meant for the intermediary device to be external to the input/output hub. Claim 8 has since been amended to recite the input/output hub *being coupled to* an intermediary device *that is external to the hub*, to better clarify the invention. Neal does not teach, suggest, or disclose the input/output hub according to claim 7, *coupled to an intermediary device that is external to the input/output hub,* the intermediary device interconnecting the Producer-Consumer ordered interface and an input/output device. Thus Applicants believe that dependent claim 8 distinguishes over the cited art and is allowable.

The Examiner rejected claims 4, 11, 13, 17, and 25 under 35 U.S.C. § 103(a) as being unpatentable over Neal in view of Cataldo. As was explained to the Examiner in the interview, the Cataldo references describes the release of the very chip set in which the present invention was incorporated. Thus it could not have been prior art. However, the Examiner still requested a Rule 131 declaration stating that the date of invention antedates the Cataldo reference. The declaration and exhibit enclosed herewith show that a draft of the present application was emailed to the inventors on August 23, 2001 at 3:23 pm PST, thus establishing the date of invention as at least August 23, 2001. The application was constructively reduced to practice four days after the draft was sent to the inventors on August 27, 2001, establishing diligence. The Cataldo reference is dated August 23, 2001. Thus, the Cataldo reference was not published *before* the Applicants' date of invention, and thus the Cataldo reference is not prior art. Based upon this, Applicants have rewritten claims 4, 11, 13, 17, and 25 in independent form and request that the rejection of claims 4, 11, 13, 17, and 25 be withdrawn.

Independent claims 5, 12, 16, and 22 recite limitations similar to independent claim 1, as amended, and thus distinguish over the cited references for the same reasons as independent claim 1, as amended. Claim 2 depends from independent claim 1 as amended, and thus distinguish over the cited references for the same reasons as independent claim 1. Claims 6-9 all depend, directly or indirectly, from independent claim 5 as amended, and thus distinguish over the cited references for the same reasons as claim 5. Claims 14-15 all depend, directly or indirectly, from independent claim 12 as amended, and thus distinguish over the cited references for the same reasons as claim 12. Claims 18-21 all depend, directly or indirectly, from independent claim 16 as amended, and thus distinguish over the cited references for the same reasons as claim 16. Claim 23 depends from independent claim 22 as amended, and thus distinguishes over the cited references for the same reasons as claim 22.

Applicants believe that the foregoing amendments place the application in condition for allowance, and a favorable action is respectfully requested. If for any reason the Examiner finds the application other than in condition for allowance, the Examiner is requested to call the undersigned attorney at the Los Angeles telephone number (213) 488-7100 to discuss the steps necessary for placing the application in condition for allowance should the Examiner believe that such a telephone conference would advance prosecution of the application.

Respectfully submitted,
PILLSBURY WINTHROP LLP

Date: May 11, 2004

By: [illegible]
Roger R. Wise (RN 31,204)
Attorney For Applicant(s)

725 South Figueroa Street, Suite 2800
Los Angeles, CA 90017-5406
Telephone: (213) 488-7100; Facsimile: (213) 629-1033

Encls: Rule 131 Declaration with Attachments.
Articles describing "Scalable Coherent Interface"

Home Contents Feedback Random Search

Scalable Coherent Interface

<*hardware*, *protocol*> (SCI) The ANSI/IEEE 1596-1992 standard that defines a point-to-point interface and a set of packet protocols. The SCI protocols use packets with a 16-byte header and 16, 64, or 256 data bytes. Each packet is protected by a 16-bit CRC code.

The standard defines 1 Gbit/second serial fiber-optic links and 1 Gbyte/second parallel copper links. SCI has two unidirectional links that operate concurrently.

The SCI protocols support shared memory by encapsulating bus requests and responses into SCI request and response packets. Packet-based handshake protocols guarantee reliable data delivery. A set of cache coherence protocols are defined to maintain cache coherence in a shared memory system.

Message passing is supported by a compatible subset of the SCI protocols. This protocol subset does not invoke SCI cache coherency protocols.

SCI uses 64-bit addressing and the most significant 16 bits are used for addressing up to 64K nodes.

http://www.uni-paderborn.de/pc2/systems/sci/.

[Applications?]

(1999-03-22)

Try this search on OneLook / Google

Nearby terms: SCADA « scag « scalability « **Scalable Coherent Interface** » Scalable Processor ARChitecture » Scalable Sampling Rate » Scalable Vector Graphics

Translate to French Powered by **SYSTRAN**

Scalable Coherent Interface

Introduction

Scalable Coherent Interface (SCI) is the modern equivalent of a Processor-Memory-I/O bus and a Local Area Network, combined and made parallel to support distributed multiprocessing with very high bandwidth, very low latency, and a scalable architecture that allows building large systems out of many inexpensive massproduced building blocks.

SCI reduces the delay of interprocessor communication by an enormous factor compared to even the newest and best interconnect technologies that are based on the previous generation of networking and I/O channel protocols (FibreChannel and ATM), because SCI eliminates the need for run-time layers of software protocol-paradigm translation. A remote communication in SCI takes place as just a part of a simple load or store opcode execution in a processor. Typically the remote address results in a cache miss, which causes the cache controller to address remote memory via SCI to get the data, and within on the order of a microsecond the remote data are fetched to cache and the processor continues execution.

To make program porting easy, the old protocols can be layered on top of SCI transparently. Of course, such an implementation only gains SCI's raw speed factor: to get SCI's full potential speedup, applications will need to eliminate the protocol overheads by using global shared memory directly.

The old approach, moving data through I/O-channel or network-style paths, requires:

- assembling an appropriate communication packet in software
- pointing the interface hardware at it
- initiating the I/O operation, usually by calling a subroutine
- when the data arrive at the destination, hardware stores them in a memory buffer
- hardware alerts the processor by an interrupt when a packet is complete or the buffers are full
- software then moves the data to a waiting user buffer (sometimes this move can be avoided, in the latest systems)
- user application examines the packet to find the desired data

Typically this process results in latencies that are tens to thousands of times slower than SCI. These latencies are the main limitation on the performance of Clusters or Networks of Workstations.

Scalable Coherent Interface

Basic Features

The SCI interface standard defines a *point to point communication* between neighbour nodes reducing the non-ideal transmission line problems of bus-based systems. It is designed to scale well as the number of attached processors increases.

The transfer rate is 1 GByte/s point-to-point. SCI allows up to *64K nodes* to be connected to an interconnect. Memory may be shared by all processors. The addressing scheme uses a 64-bit Fixed Addressing Model. It uses 48 bit as a node offset address and 16 bit for the node addressing.

SCI defines an interface standard that enables the use of many different interconnect configurations ranging from simple rings to complex multistage switching networks. The SCI-Protocols define a set of packet based bus transactions. They can very effectively be used in clustered systems because they have been designed for distributed processing and are based on an underlying multiprocessor architecture. The protocol use small packets (16-288 bytes including header) to carry data and commands between nodes with low latency. Packets can be pipelined to achieve high overall transfer rate.

Basic features

- shared memory programming model
- directory based cache coherence protocol
- distributed directory scheme
- 64K addressable nodes; 16 bit node identifier
- Split phase transaction protocol
- support for non-switched and switched topologies
- Packet switching network, ring or crossbar
- physical level definition
- logical level definition
- C language model of protocols
- 1 GByte/s 16bit parallel; 1Gbit/s serial

Scalable Coherent Interface

Implementations

SCI was completed in 1991, and became an approved ANSI/IEEE standard in 1992.

Computers using SCI:

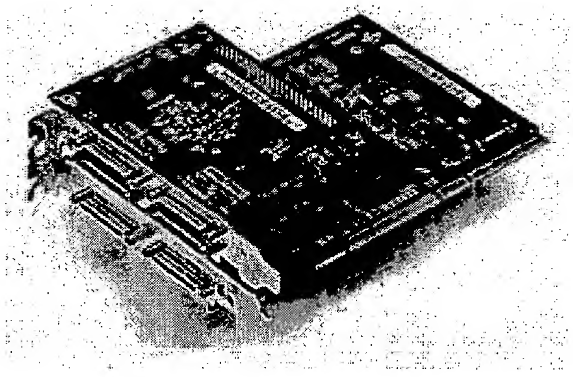
- *Hewlett-Packard/Convex Exemplar SPP-2000*
  (Shared memory, PA/RISC based)
  http://www.top500.org/ORSC/1997/exemplar.html
- *National Supercomputer Centre (NSC) Monolith*, Sweden/2002
  200x Dual XEON 2.2GHz w/t Dolphin/SCALI SCI
  TOP500 11/03 rank: 103
  http://www.nsc.liu.se/systems/monolith/

Implementations by:

- Dolphinics, www.dolphinics.com

References:

- see above
- Hellwanger, Reinefeld: SCI: Scalable Coherent Interface
- Hwang, Xu: Scalable Parallel Computing



SCI card (3D) by Dolphinics

Scalable Coherent Interface

Transactions and Protocols

Communication among nodes is accommodated by a set of SCI transactions and protocols that include support for:

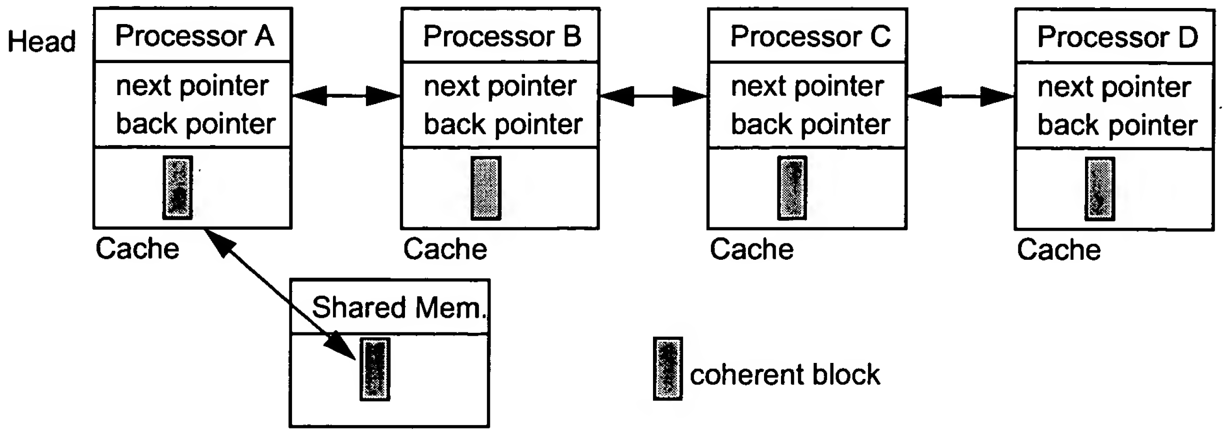
- data read/write
- cache coherence
- synchronization primitives
- message passing

All transactions are sent as SCI packets between soure and destination nodes. Protocols are provided to handle flow control, error recovery, and deadlock prevention. The transaction format definition is independent of the network topology.

SCI uses a distributed directory-based cache coherence protocol. Each shared line of memory is associated with a distributed list of processors sharing that line. All nodes with cached copies participate in the update of this list.

The shared cache lines are linked together by a double linked list. The coherence list pointers are the nodes addresses. The typical cache line size is 64 bytes.

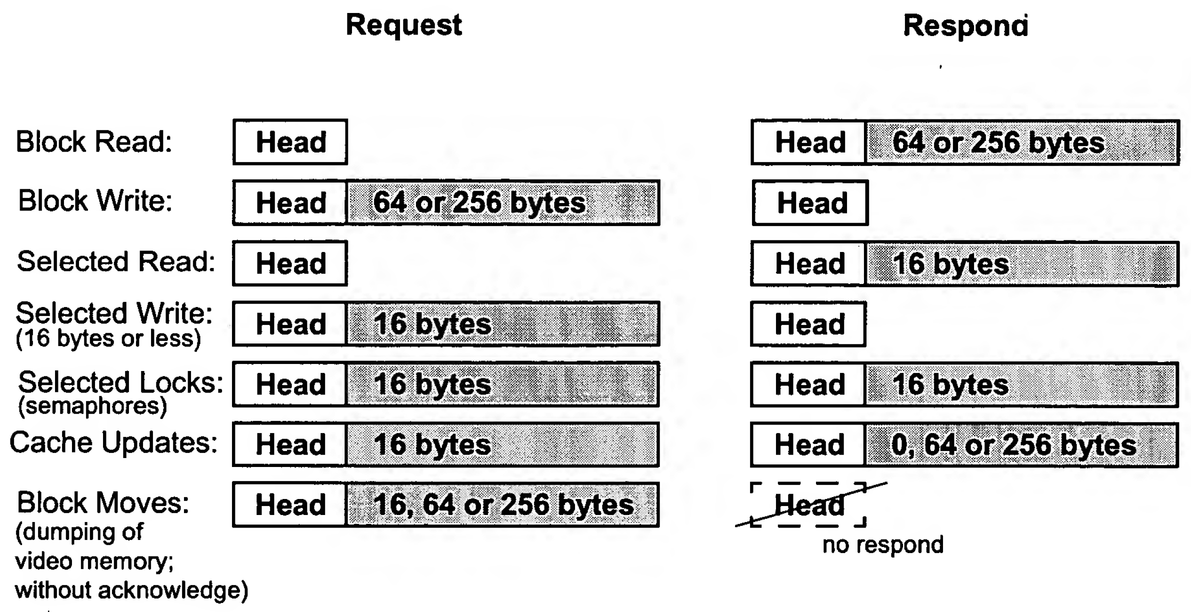
The cache coherence transaction manipulate a linked-list structure used to maintain a coherent memory image.



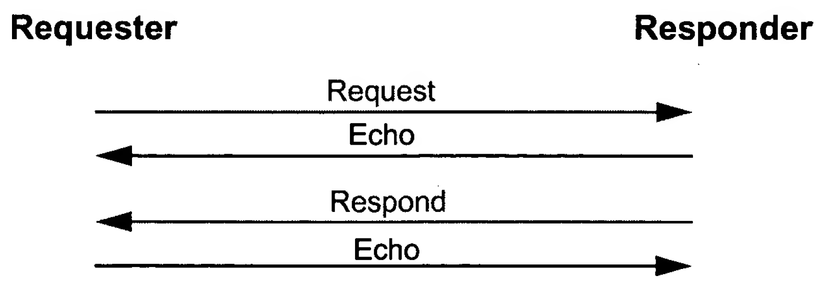
Directory structure

Scalable Coherent Interface

Transaction Formats



Transaction Formats



Scalable Coherent Interface

Synchronization Primitives

SCI supports two types of lock primitives, cached and non-cached.

- *Cached locks* are implemented by temporarily locking a cache-line in the only_dirty (exclusive modified) state while these instruction sequences are executed.
- *Non-cached lock transactions* (e.g. swap, swap-and-compare) are provided for accessing non-cached shared data.

An important feature of SCI is the ability to interface to other busses. A bus converter can simply translate the SCI commands to native bus cycles because they are very similar. Two cases are handled with special care, *bus locking and cache coherence*. Most backplane busses have a atomic read-modify-write operation to manipulate semaphores and other critical data. Since SCI is a four phase transaction protocol with no guaranteed delivery in order, the lock is defined as a single SCI command.

SCI supports message passing as defined by IEEE Std. 1212. A standard non-coherent write64 transaction is used to send short messages to a specified control and status register (CSR) within the target node. A typical application for such transaction is accessing control registers of an I/O bridge.

Passing of interrupts between CPU nodes and/or I/O bridges is supported by a standard SCI write transaction to a CSR register.

Scalable Coherent Interface

Physical level

Signaling

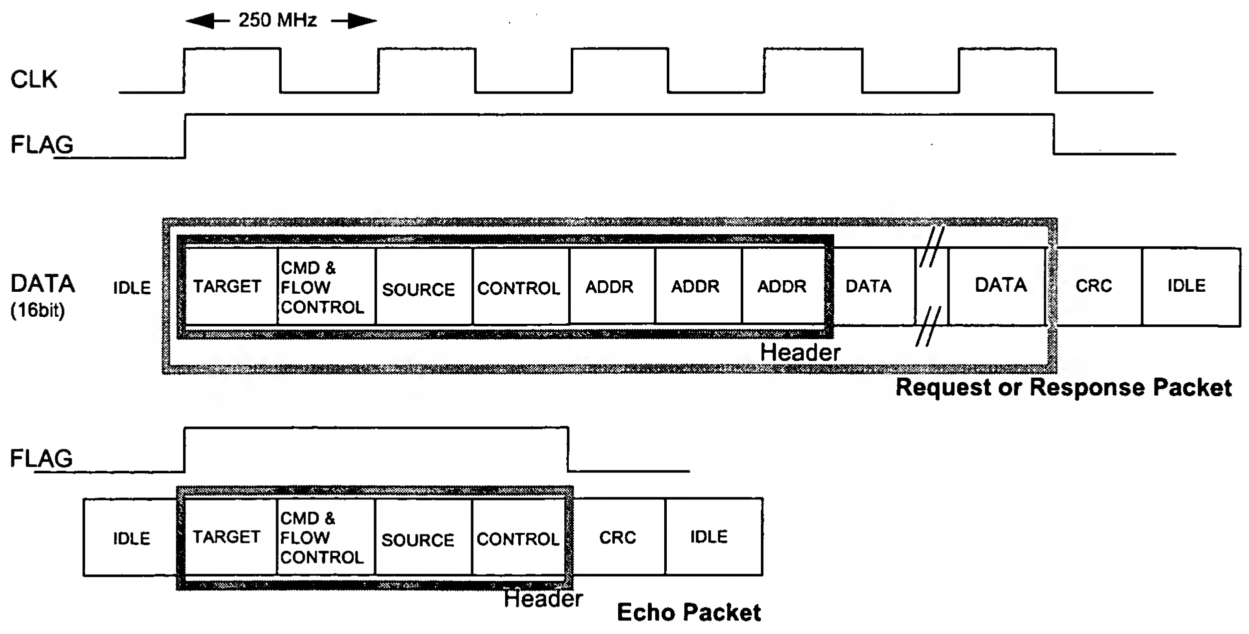
SCI uses a narrow 16-bit data path (plus clock and one flag bit) at 2ns/word (250MHz clock, both edges active), to controll the interface chip pin-count problem and make switch elements more practical. The signals are transfered using differential ECL voltage levels. One unidirectional SCI connection requires 36 signal wires. The complete bi-directional SCI link sums up to 72 signal wires.

Packets

A packet consists of three main sections:

- header section
- address and data section
- error check symbol

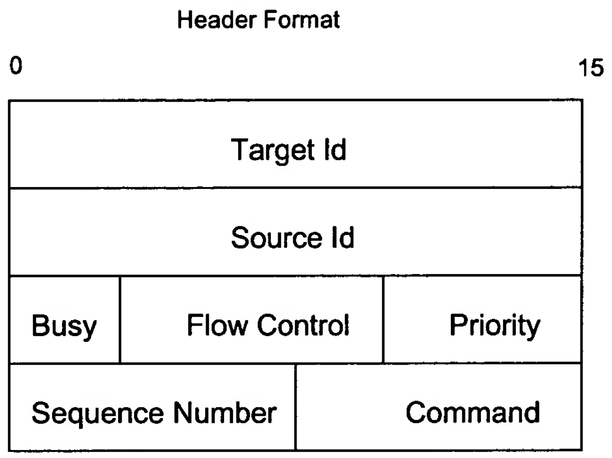
The first word of the paket contains the ID code of the final receiving node, the target. By looking at the first word of the packet, a node or switch can quickly determine where to route this packet.



Packet format

Scalable Coherent Interface

Physical level



Header format

The control word of the header controls packet flow and network access. Priority arbitration is supported with round robin arbitration on each level.

The command word of the header contains the transaction command and a sequence number. The sequence number is a tag to identify a packet.

A node may send many requests (up to 256; 64 ?) before a response is received. This transaction pipeline can cause responses to returned out-of-order, and therefore a sequence number is needed to identify a response with the corresponding request.

The command field contains the command a receiver must execute.

Scalable Coherent Interface

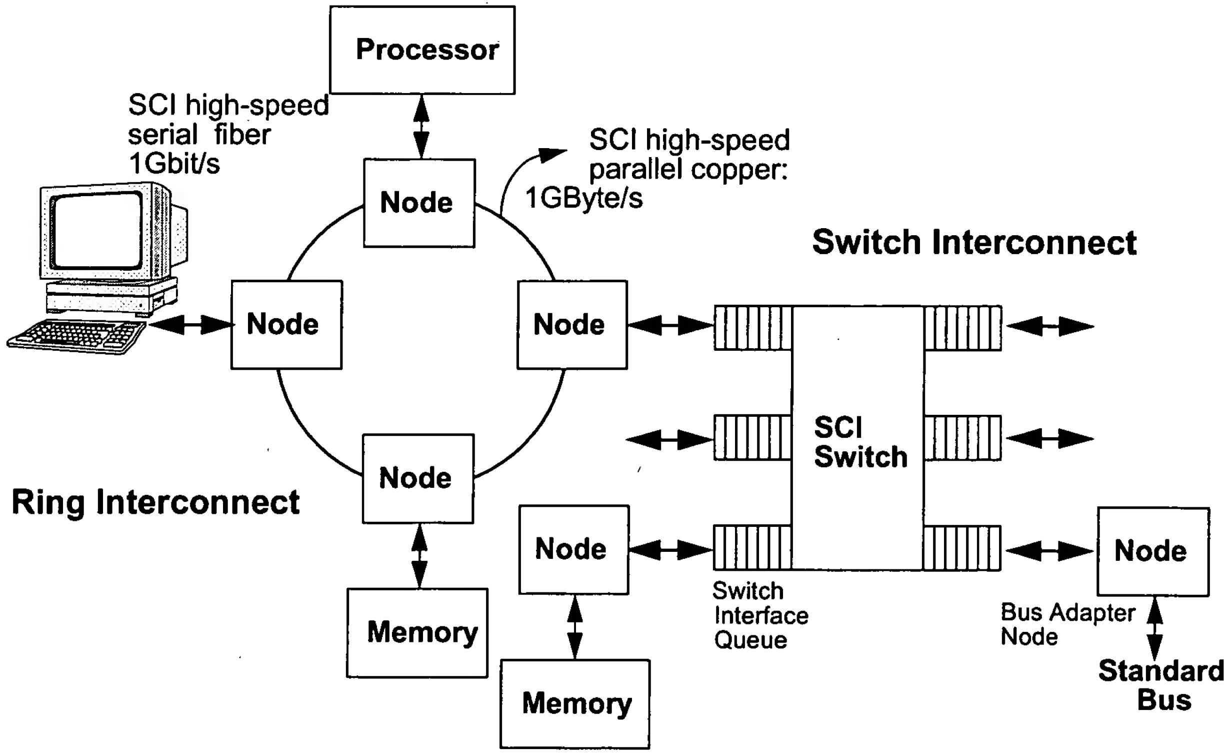
Topology

The ring interconnect is the simplest structure. There, nodes pass packets to their neighbour. In such a structure are no active components except the nodes. The nodes have to control the arbitration, priority and forward progress schemes.

A switched solutions is provides more perfomance:

Modular Switch for System Area Networks from Dolphin

- Scalable switch (standard = 4 nodes) supports clusters with up to 16 nodes
- 6.4 Gigabit/sec links (bidirectional) provide high data throughput
- Ultra-low port-to-port packet routing latencies
- High availability clusters supported by hot-pluggable ports, port fencing, and redundant links
- ANSI/IEEE 1596-1992 Scalable Coherent Interface (SCI) compliant
- Non-blocking, Cut-through routing

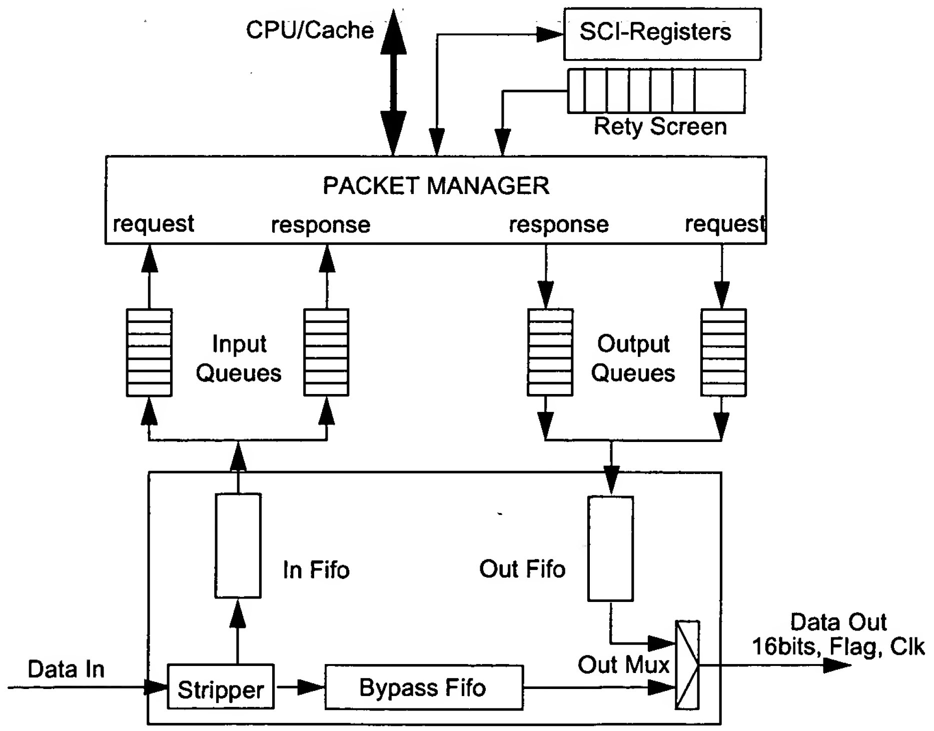


Topology example

Scalable Coherent Interface

Node interface

An SCI node receives a steady stream of data and transmit another stream of data. These streams consists of SCI packets and IDLE symbols. A node that is granted interconnect access and that has an empty bypass fifo is allowed to transmit a packet. Contention is solved either by buffering in the interconnection network or by filling the bypass fifo of the next node. SCI uses idles, packet headers, and echos to selectively grant interconnect access under heavy system loading.



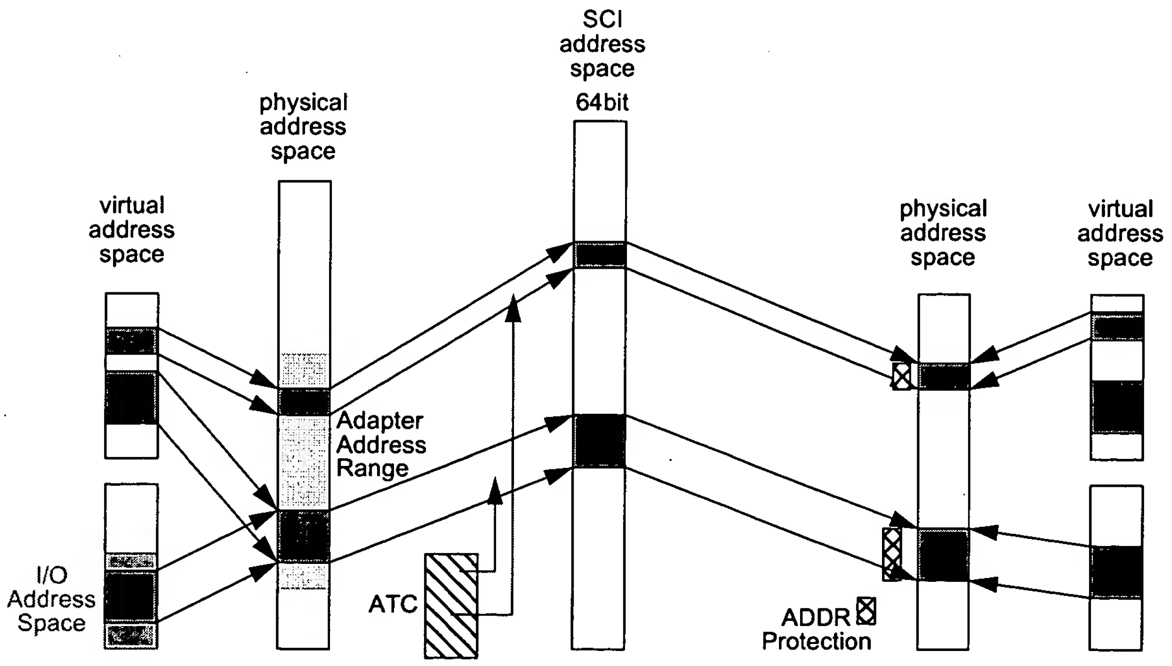
Node interface

Scalable Coherent Interface

Address Map

The PCI-SCI card translates a a 32-bit PCI address into a 64-bit SCI address and vice versa. The outgoing translation is based on address translation tables residing in PCI memory. Address translation table entries are cached in the PSB address translation cache. For incoming SCI requests, address protection can be used to disable access to the node or certain PCI memory regions.

In order for a processor to directly access far memory through load/store instructions, the far memory must be addressed through an SCI address. A 64-bit SCI address is used to select nodes and address data within a node. The 64-bit SCI address is split into a 16-bit node id that is used to select the target node. The remaining 48 bits are used within a node to address data and CSR registers.



Address map

Scalable Coherent Interface

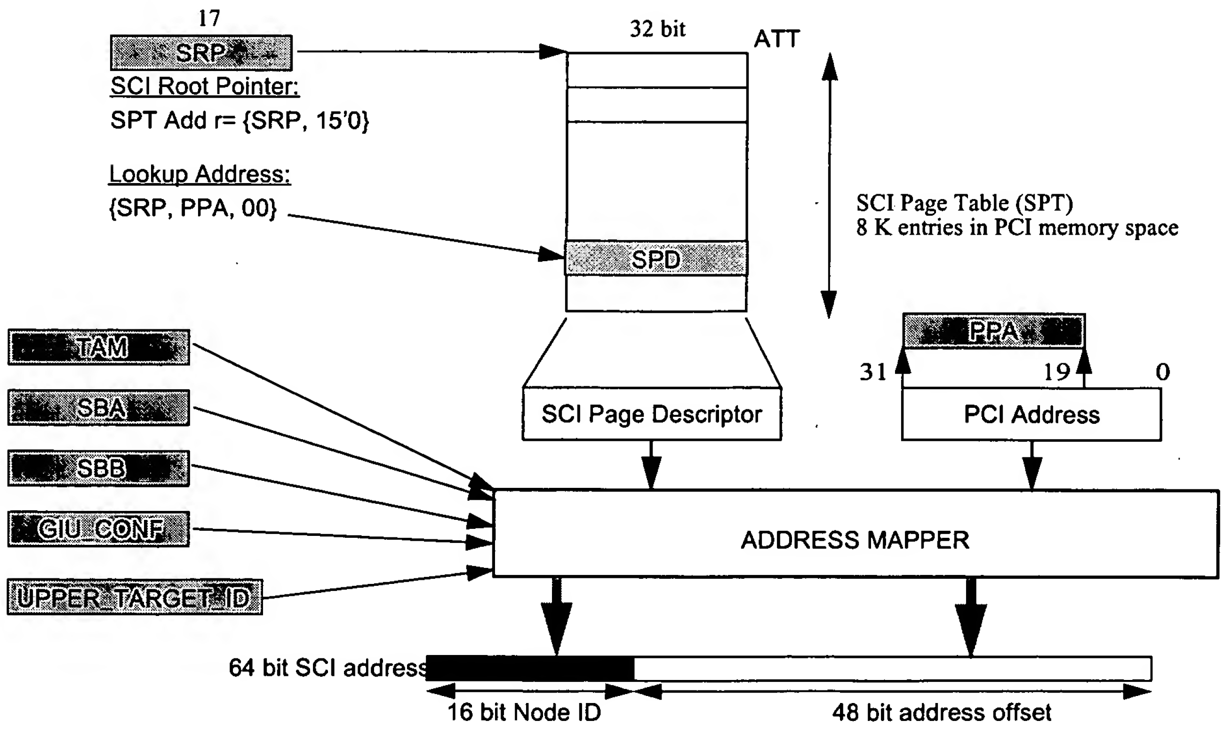
Address Mapper

During system configuration, driver software on each of the nodes will agree upon (using messages) the desired mappings and set up the contents of the page tables. Each node has a page table that allows that nodes map memory from one or more other nodes. Once the page tables have been configured, the processor can access far memory through load/store instructions.

The address translation is implemented using an address translation table (ATT). The page table consists of 8K entries of 32-bits each. The page size is 512 KBytes.

The information contained in a page table entry are:

- *Node ID*
- *Node Offset Adress*
- *Valid*: Shows if the current entry is valid
- *Atomic* (Lock bit): An access will generate an atomic operation (FETCH_ADD+1)
- *Ordering*: Enforces write before read ordering.



Address mapper